ATM Simulation Links - by Topic

Introduction

[Introduction]

Requirements

[Requirements]

Analysis

[Use Cases]
[Initial Functional Tests]
[Analysis Classes]

Design

[CRC Cards]
[Class Diagram]
[State Charts]
[Interaction Diagrams]

Detailed Design and Implementation

[Detailed Design]
[Package Diagram]
[Code]

Testing

[Executable Applet]

Maintenance

[Maintenance Ideas]

[Links by Class]

Author and Copyright

Introduction

What you will Find Here

This page is the starting point into a series of pages that attempt to give a complete example of object-oriented analysis, design, and programming applied to a moderate size problem: the simulation of an Automated Teller Machine. I developed these pages in the belief that students would benefit from seeing a *complete* example of OO methodology applied to a single problem. Since then, I have developed a similar-style solution to an even simpler problem: maintaining a *very* simple address book.

Beginning with a statement of requirements, the process proceeds through analysis, overall design, and detailed design and implementation, culminating with some suggestions for maintenance. (Testing is left as an exercise to the reader :-).)

Analysis is done by identifying the use cases and detailing a flow of events for each. Also, an initial set of functional test cases is specified, to serve as a vehicle for checking that the implementation is complete and basically correct. Analysis culminates in identifying classes implied by the use cases, and documenting them using an Analysis Class Diagram. (The Statechart diagrams done under design are also, in part, analysis tasks. In OO, the line between analysis and design is not always a sharp one.)

Overall design begins by using CRC cards to assign responsibilities to the various classes. The static structure of the design is summarized by means of an overall Class Diagram. Then the dynamic aspects of the design are developed, using State Charts for the major controller classes, plus an Interaction Diagram for each of the main use cases.

The detailed design is developed by spelling out the attributes and methods for each class, using a class diagram for each class with all three "compartments" fully filled in. A package diagram is used to show how the various classes are groupled into packages. Each class is then implemented in Java. The code page contains links both to Javadoc documentation for each class, and to the complete source code. Also included are a main class, which allows the simulation to run as an application, and an Applet class, which allows it to be run as an Applet. (This illustrates how the same basic code can be designed to be used either way; the application was used for most of the initial development work, but the Applet is accessible to anyone over the web).

The Executable Applet version can be loaded and executed from within any web browser that supports at least JDK 1.1.x.

The Maintenance page discusses ideas for possible changes to the system, each of which would require changes to multiple documents, not just the code.

Background

This is the second significant revision of this series of pages.

1. I originally developed this series of pages for use in a course which was taught to junior CS majors at Gordon College from 1996 to 1999, using C++. (That version is available here - from which you can follow a link to an even older version if you wish.)
2. The second version represents a major revision (done during 2000 and 2001) for use in "Object-Oriented Software Development", which is taught to sophmore CS majors and uses Java. The most signficant change in the second and subsequent versions is the use of UML notation throughout, together with a reorganization of the process to more closely follow the UML approach. Moreover, the original version was implemented in both C++ and Java; but subsequent versions uses only Java, so as to be able to utilize some features of that language (e.g. threads) which have no direct analog in the C++ language per se. That version is available here.
3. I more recently (in 2002) revised the sequencing of the pages to reflect a slightly different ordering of the steps in the software development process as I am now teaching it in 2002. (The content of the current set of pages is almost exactly the